[0026] Transposition (t) hash: OCAT, CAOT, COTA

[0027] Deletion (d) hash: OAT, CAT, COT, COA

[0028] Transposition-replacement (tr) hash: *CAT, O*AT, OC*T, OCA*, *AOT, C*OT, CA*T, CAO*, *OTA, C*TA, CO*A, COT*

[0029] Special deletion-transposition hash: ACT, CTO

[0030] Double deletion hash: CO, CA, CT, OA, OT, AT

[0031] Deletion-replacement hash: *AT, O*T, OA*, C*T, CA*, *OT, CO*, *OA, C*A

[0032] Insertion-replacement hash: **OAT, *C*AT, *CO*T, *COA*, C**AT, C*O*T, C*OA*, *O*AT, CO**T, CO*A*, *OA*T, C*A*T, COA**, *OAT*, C*AT*, CO*T*

[0033] These hash tables require, in total $O(D*W^2)$ storage.

[0034] Referring again to FIG. **3**, with the above hash tables pre-created, a test is performed for a distance one misspelling (as discussed above in conjunction with FIG. **1**). As noted earlier, this takes O(W) time and requires storage that is O(D)=O(D*W). If the distance one misspelling routine indicates that the candidate word is spelled correctly, the algorithm terminates, also indicating a correct spelling (Step **320**). Otherwise, it checks to see if enough suggested corrections have been accrued in testing for a distance one correction (Step **330**). If enough have been detected it outputs the suggested corrections and terminates (Step **340**), otherwise it goes through the process of testing distance two variants of the candidate word against the distance two hash tables (Step **400**), and only then outputs suggested corrections and terminates (Step **340**).

[0035] FIG. **4** is a flow chart illustrating the process **400** of testing variants of the candidate word against hash tables derived from the dictionary for distance two misspellings. If a transposition is denoted by t, a deletion by d, an insertion by i, and a replacement by r; the following misspellings are possible: tt, td, ti, tr, dt, dd, di, dr, it, id, ii, ir, rt, rd, ri, rr.

[0036] The following table lists the misspelling type, the action, and the hash table checked for each of the 16 possible distance two misspellings. Note that the possible outcomes of two successive misspellings xy, where x,y are elements of {t,d,i,r} are the same as the successive misspellings of yx, except in the single case where td # dt, since for example, on the one hand, starting with the word COAT one can reach CTO via a transposition followed by a deletion, but not via a deletion followed by a transposition, and on the other hand, starting from COAT one can reach OAT via a transposition followed by a deletion but not vice versa. Note that there is also an asymmetry in it and ti, where, for example (again from the word COAT) the CO*AT variant is not obtainable from ti and the ti variant CA*OT is not obtainable from it. However, the first of these variants is caught in a distance one simple insertion check, so can be disregarded, and the second variant is caught just like all other it or ti variants by the d Test Action against the t hash table. Only in the two cases, of td and dt are two separate actions followed by hash table checks required. The dt hash is a special hash since it does not need to store all deletions followed by transpositions, since most of these will be caught by the t test action against the d hash table. The exceptional cases are those where one first deletes a character and then transposes the characters that were originally around the deleted character. Only these O(W) deletion-transpositions need to be stored in the dt hash table.

TABLE 1

Misspelling Type	Test Action	Hash table checked
tt	t	t
td	t	d
	(none)	d
ti	d	t
tr	r	tr
dt	t	d
	(none)	dt
dd	(none)	dd
di	d	d
dr	r	dr
it	d	t
id	d	d
ii	dd	dictionary
ir	r	ir
rt	r	tr
rd	r	dr
ri	r	ir
rr	dd	dd

[0037] Returning to FIG. **4**, after starting (Step **405**) and obtaining the candidate word, one tests the candidate word against the deletion, deletion-transposition and double deletion hash tables, and accumulates matches in Step **410**, as verified by checking Table 1. Next, one generates all single move transpositions of the candidate word in Step **415** and tests these variants against the transposition and deletion hashes in Step **420**. Next, in Step **425** one generates all single character deletion variants and in Step **430** tests these against the transposition and deletion hashes. Note that just the various test actions that go against the same hash tables in Table 1 are accumulated and being executed in a single step, for the sake of brevity of explanation. Next, in Step **435** one generates all single character replacement variants of the candidate word and in Step **440** tests against the translation-replacement, deletion-replacement, and insertion-replacement hash tables. Finally, in Step **445** all double deletions of the candidate word are generated and in Step **450** these are tested against the double deletion hash. Having accumulated all hash table matches, the results are output in Step **455**.

[0038] It is noted that except for distance two misspellings that involve double insertions, double deletions, or replacements, all actions can be done in O(W) time with O(D*W) =O(D) storage. However, replacements are less usual than the other single operations, and may be considered to be a deletion followed by an insertion. Also, double insertions and double deletions are relatively rare types of misspellings. Hence, if distance two misspellings are re-defined to exclude these possibilities (i.e., don't test these cases), a correction algorithm is provided that runs in O(W) time with O(D*W)=O(D) storage. This is referred to as a "soft" distance two correction.

[0039] FIG. **5** is a flow chart illustrating the overall flow of the "soft" distance two spelling correction algorithm **500**. The diagram of FIG. **5** is identical to FIG. **3**, the complete distance two correction flow, except that in lieu of testing all distance two variants against the relevant distance two hash tables, only distance two variants are tested that do not include a replacement, and do not include double deletion and double replacement. As usual, relevant hash tables are assumed to be pre-created. Distance one correction is initially performed (Step **100**). If no misspelling is detected, the process **500** outputs that the word is spelled correctly and terminates (Step **520**). Otherwise, a test is performed to see